Etape 1bis

- API NIO 2 : depuis JAVA 7
- sources :
  - java oracle documentation
  - Java NIO, R. Hitchens, O'Reilly ed.

- le package **java.nio** (new io.2) apporte de nouvelles fonctionnalités (complémentaires) aux classes IO originelles
  - *améliorations sur l'implémentation (plus rapide)*
  - *sur l'API : nouveaux types (davantage abstraits) et utilitaires pour:*
    - *les interactions avec le File system : Path, Paths Files, FileSystem, …*
    - *lecture / écriture non bloquantes : Channels, Buffers, Selectors*
  - *l'encodage des caractère : java.nio.Charset (critique dans certaines situations)*

- ***Path*** *: interface qui représente un chemin ou sous chemin (fichier, répertoire, sous chemin, …)*
  - *Paths : un utilitaire qui permet aussi de créer des Path*

- ***Path (java.nio.file.Path)*** *: s'agissant d'une interface la création des Path passe par l'utilitaire FileSystems ou Paths qui fournissent des fabriques à Path*

```
Path path1 = FileSystems.getDefault().getPath("logs", "access.log");
Path path2 = Paths.get("C:/Users/bobby/AppData/Local/logs/access.log");
```

=> existe tout une api sur *le chemin* : `getFileName(),getRoot(), getParent(),getName(index)` qui retourne un élément du chemin en fonction de sa position

=> Path objet *iterable* donc on peut le parcourir pour accéder aux différente parties du chemin :

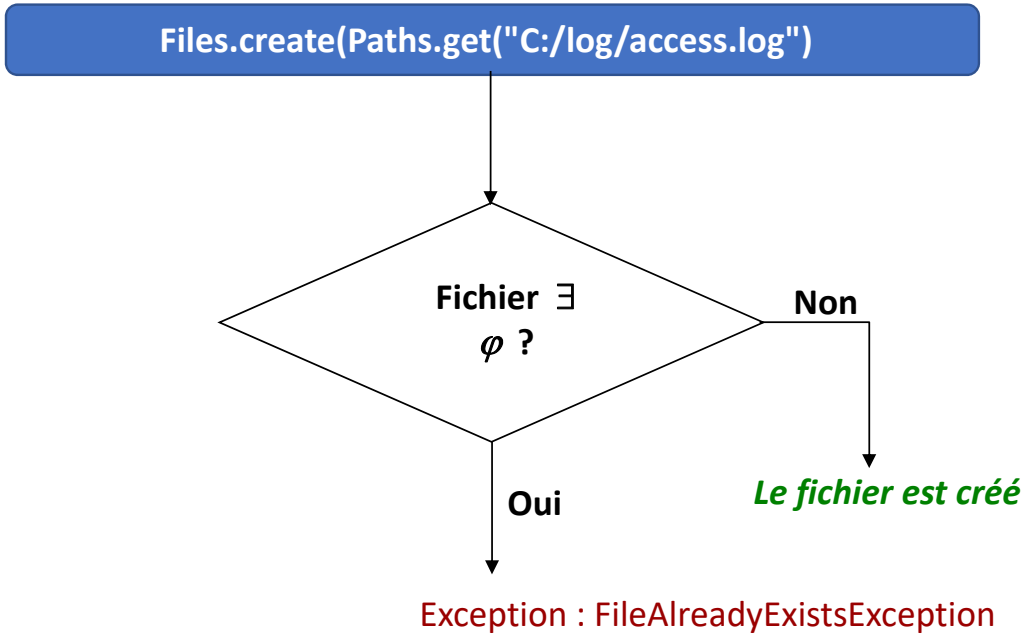
```
Path path = Paths.get("C:/Users/bobby/AppData/Local/Temp/access.log");
    for (Path morceaux : path) {
      System.out.println(morceaux);
    }
```

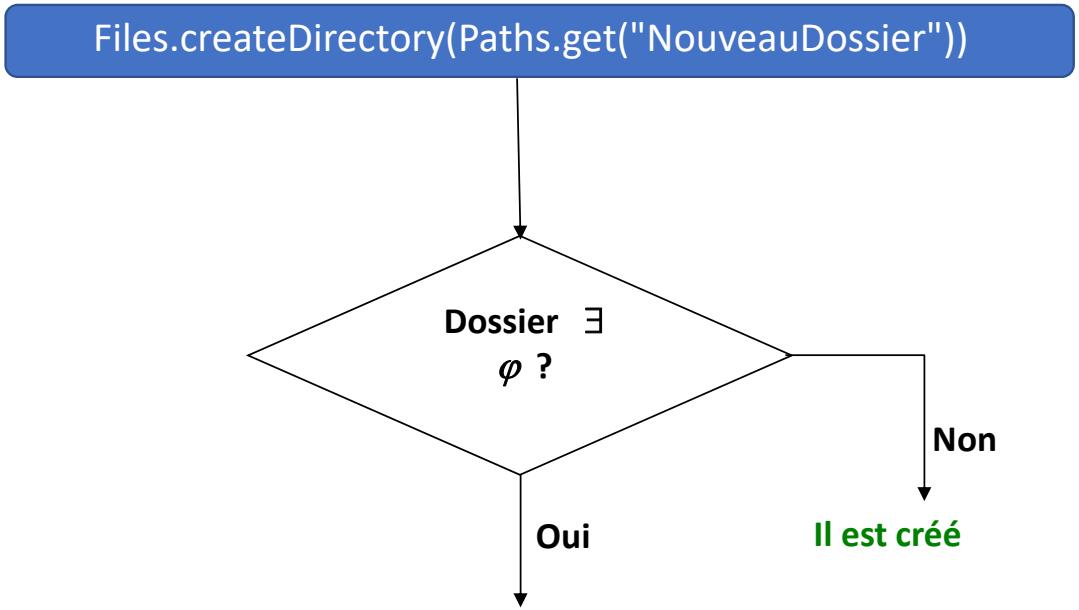
- ***Files (java.nio.file.Files)*** *:* apporte des méthodes static de plus haut nouveau d'abstraction

  - *copy(), move(), delete(), createDirectory(), createFile(), createLink(), createSymbolicLink()…*
  - pour remplacer notamment des opérations qui auraient nécessité plusieurs instructions avec l'api io

  - ***File System*** *: gérer le File System sous-jacent ou concevoir un nouveau au dessus d'un FileStorage*
    - *on s'abstrait encore davantage des particularités de l'OS (root, séparateur, …)*

- Le fichier ou le répertoire correspondant à un **Path** peut exister ou ne pas exister.
  On peut le manipuler ainsi (ajouter des morceaux, extraire des parties, le comparer à d'autres …) puis utiliser des méthodes de **Files** pour vérifier l'existence, créer, supprimer changer les permissions …




Files.createDirectory(Paths.get("NouveauDossier"))
Dossier ∃ φ ?
Non
Il est créé
Oui
Exception :
java.nio.file.FileAlreadyExistsException

- ***Passerelles entre IO et NIO :***
  - la méthode ***toPath()*** de la classe the **java.io.File** retourne l'objet **Path** correspondant
  - la méthode ***toFile()*** de l'interface Path renvoie l'objet **File** correspondant